International Journal of Research and Applications (July - Sep © 2014 Transactions) 1(3): 115-119
International Conference on Emerging Trends in Engineering & Technology (ICETET-2014)
ISSN (online): 2349-0020
ELECTRONICS ORIGINAL ARTICLE

Low Power Multiplier to Reduce Switching Activities Using Bypassing Technique

A Deepthi [1], T Santhosh Kumar [2] and Dr. P Bhaskar Reddy [3]
[123] Department of ECE, MLR Institute of Technology, Dundigal, Hyderabad, India
[1] amaram.deepthireddy@gmail.com, [2] santoshkumar.tula@gmail.com and [3] pbhaskarareddy@rediffmail.com

ABSTRACT

Multipliers and adders are the basic circuits required for implementing any Arithmetic and logic functions in VLSI. Many of the real-time applications like the arithmetic operations in Microprocessor, the filter designing in Signal processing require the multipliers. As the multipliers play a major role in the VLSI designing the power consumption related to them is a parameter to be thought of. To achieve such power reduction, modifications are made to conventional multiplier architecture and low power multiplier architecture is proposed using a technique called bypassing in this work. This proposed architecture removes the unnecessary switching activities responsible for power consumption and also eliminates unnecessary iteration done in the conventional multiplier whenever a zero is encountered in the multiplier.

Index Terms — low power, multiplier, shift-and-add, ring counter, feeder.

INTRODUCTION

As the scale of integration keeps growing, more and more sophisticated signal processing systems are being implemented on a VLSI chip. These signal processing applications not only demand great computation capacity but also consume considerable amount of energy. For portable applications where the power consumption is the most important parameter, one should reduce the power dissipation as much as possible. One of the best ways to reduce the dynamic power dissipation is to minimize the total switching activity, i.e., the total number of signal transitions of the system.

Multiplier is one of the most important arithmetic circuits used in multimedia, and digital signal processing such as discrete cosine transform, fast Fourier transform. Because of its massive usefulness lots of algorithms are developed to improve constraints like power and speed. Multiplication consists of three major steps:

1) Partial products recoding and generating;
2) Reducing the partial products by partial product reduction schemes to two rows; and
3) Adding the remaining two rows of partial products by using a carry-propagate adder to obtain the final product.

In this work, we propose modifications to the conventional architecture of the shift-and-add radix-2 multipliers to considerably reduce its energy consumption.

A. Conventional Shift and Add Multiplier: Conventional shift-and-add multiplier, which multiplies A by B architecture is shown in FIG1.There are six major sources of switching activity in the multiplier. These sources, which are marked with ovals in the figure, are:

(1) shifting value of B register,
(2) counter activity,
(3) adder activity,
(4) 0 and A switching between in the multiplexer,
(5) multiplexer select activity,
(6) partial product shifting.

Note that the activity of the adder consists of required transitions (when B(0) is nonzero) and unnecessary transitions (when B(0) is zero).

By minimizing any of these switching activity sources, it can lower the power consumption. Since some of the nodes have higher capacitance, reducing their switching will lead to more power reduction. As an example, the selector line of the multiplexer B(0) which is connected to K gates for a K-bit multiplier. If we somehow eliminate this node, power saving can be achieved. Next, we describe how we minimize or possibly eliminate these sources of switching activity.

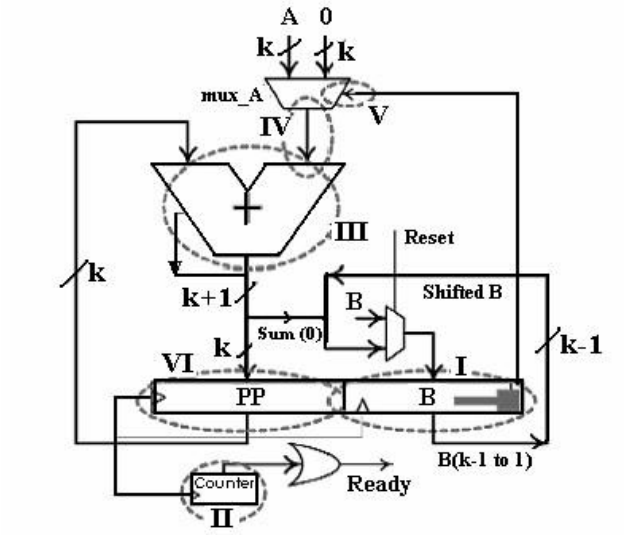


Figure1: Architecture of conventional shift and add multiplier with major source of switching activity

I. The Proposed Low Power Multiplier: BZFAD

To derive low-power architecture, we concentrate our effort on eliminating or reducing the sources of the switching activity discussed in the previous section. The proposed architecture which is shown in Figure 2 is called BZ-FAD.

A. Shift of the B Register

In the traditional architecture (see Fig. 1), to generate the partial product, B (0) is used to decide between A and 0. If the bit is "1", A should be added to the previous partial product, whereas if it is "0", no addition operation is needed to generate the partial product. Hence, in each cycle, register B should be shifted to the right so that its right bit appears at B(0); this operation gives rise to some switching activity. To avoid this, in the proposed architecture (see Fig. 2) a multiplexer M1 with one-hot encoded bus selector chooses the hot bit of B in each cycle. A ring counter is used to select B (n) in the nth cycle. As will be seen later, the same counter can be used for block M2 as well. The ring counter used in the proposed multiplier is noticeably wider (32 bits versus 5 bits for a 32-bit multiplier) than the binary counter used in the conventional architecture; therefore an ordinary ring counter, if used in BZ-FAD, would raise more transitions than its binary counterpart in the conventional architecture. To minimize the switching activity of the counter, we utilize the low-power ring counter.

To avoid this, in the proposed architecture (Fig 2) a multiplexer (M1) with one-hot encoded bus selector chooses the hot bit of B in each cycle. A ring counter is used to select B(n) in the nth cycle. As will be seen later, the same counter can be used for block M2 as well. The ring counter used in the proposed multiplier is noticeably wider (32 bits vs. 5 bits for a 32-bit multiplier) than the binary counter used in the conventional architecture; therefore an ordinary ring counter, if used in BZ-FAD, would raise more transitions than its binary counterpart in the conventional architecture. To minimize the switching activity of the counter, we utilize the low-power ring counter.

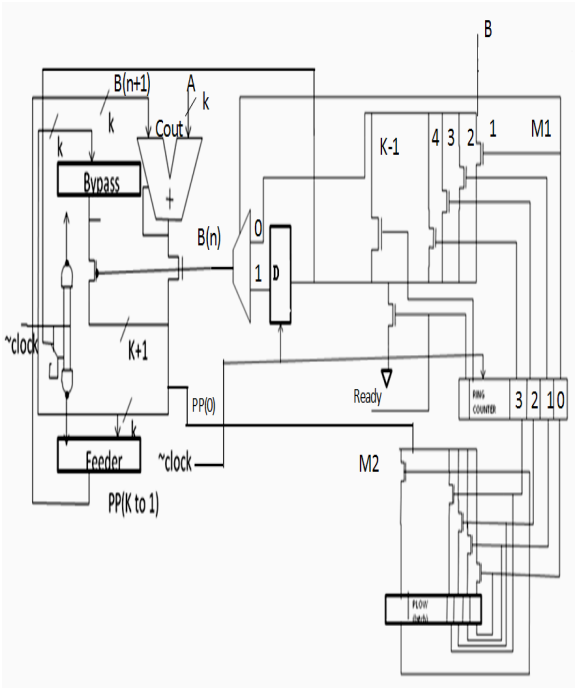


Fig 2: BZFAD Architecture

II. Shifting of PP Register

In the conventional architecture, the partial product is shifted in each cycle giving rise to transitions. Inspecting the multiplication algorithm reveals that the multiplication may be completed by processing the most significant bits of the partial product, and hence, it is not necessary for the least significant bits of the partial product to be shifted. We take advantage of this observation in the BZ-FAD architecture. Notice that in FIG 2 for Plow the lower half of the partial product, we use _ latches. These latches are indicated by the dotted rectangle M2 in Fig. 2. In the first cycle, the least significant bit PP (0) of the product becomes finalized and is stored in the right-most latch of Plow. The ring counter output is used to open (unlatch) the proper latch. This is achieved by connecting the S/~H line of the nth latch to the nth bit of the ring counter which is "1" in the nth cycle. In this way, the nth latch samples the value of the nth bit of the final product (see Fig. 2).

In the subsequent cycles, the next least significant bits are finalized and stored in the proper latches. When the last bit is stored in the left-most latch, the higher and lower halves of the partial product form the final product result. Using this method, no shifting of the lower half of the partial product is required. The higher part of the partial product, however, is still shifted. Comparing the two architectures, BZ-FAD saves power for two reasons: first, the lower half of the partial product is not shifted, and second, this half is implemented with latches instead of flip-flops. Note that in the conventional architecture (see Fig. 1) the data transparency problem of latches prohibits us from using latches instead of flip-flops for forming the lower half of the partial product. This problem does not exist in BZ-FAD since the lower half is not formed by shifting the bits in a shift Register.

In brief, from the six sources of activity in the multiplier, we have eliminated the shift of the B register, reduced the activities of the right input of the adder, and lowered the activities on the multiplexer select line. In addition, we have minimized the activities in the adder, the activities in the counter, and the shifts in the PP (partial product) register. The proposed architecture, however, introduces new sources of activities. These include the activities of a new multiplexer which has the same size as that of the multiplexer of the conventional architecture.

III. Reducing Switching Activity of the Adder:

In the conventional multiplier architecture (see Fig. 1), in each cycle, the current partial product is added to A (when B (0) is one) or to 0 (when B (0) is zero). This leads to unnecessary transitions in the adder when B (0) is zero. In these cases, the adder can be bypassed and the partial product should be shifted to the right by one bit. This is what is performed in the proposed Architecture which eliminates unnecessary switching activities in the adder.

As shown in Fig. 2, the Feeder and Bypass registers are used to bypass the adder in the cycles where B (n) is zero. In each cycle, the hot bit of the next cycle (i.e., B (n+1)) is checked. If it is 0, i.e., the adder is not needed in the next cycle, the Bypass register is clocked to store the current partial product. If B(n+!) Is 1, i.e., the adder is really needed in the next cycle, the Feeder register is clocked to store the current partial product which must be fed to the adder in the next cycle. Note that to select between the Feeder and Bypass registers we have used NAND and NOR gates which are inverting logic, therefore, the inverted clock (~ Clock in Fig. 3) is fed to them. Finally, in each cycle, B (n) determines if the partial product should come from the Bypass register or from the Adder output.

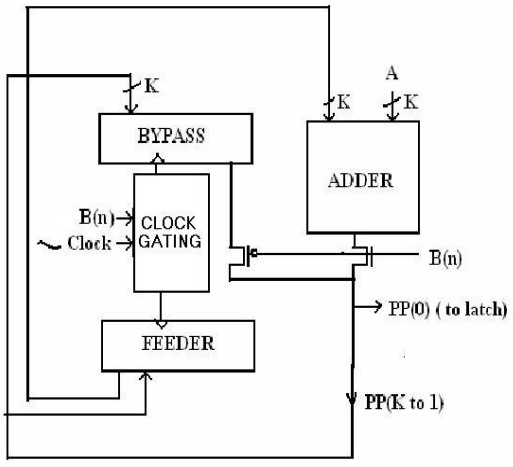


Fig 3.Adder with Registers

Feeder and bypass registers are used to optimize the adder operation. In every cycle, the partial product obtained in the previous cycle is available at the input of these two registers. A clock gating structure is given to feeder and bypass registers. The clock gator performs the following functions. Feeder is clocked if LSB of 'B' obtained by using the low power ring counter is equal to '1'. Bypass is clocked if LSB of 'B' obtained by using the low power ring counter is equal to '0'. Thus the current partial product is stored either in feeder or in bypass register.

IV. Critical Path Analysis

By comparing the two architectures, it is observed that in BZ-FAD, the value of B does not contribute to the critical path whereas in the conventional multiplier, it should first settle since B(0) is required for the multiplexer to select either A or zero. In BZ-FAD, "A" has already settled and only the output of the Feeder register which is the other input to the adder needs to settle. Next, we consider the BZ-FAD MUX1, whose select signal does not contribute to the critical path. This is because the adder in the BZ-FAD, in contrast to the conventional architecture, can begin its work independent of multiplexer MUX1.

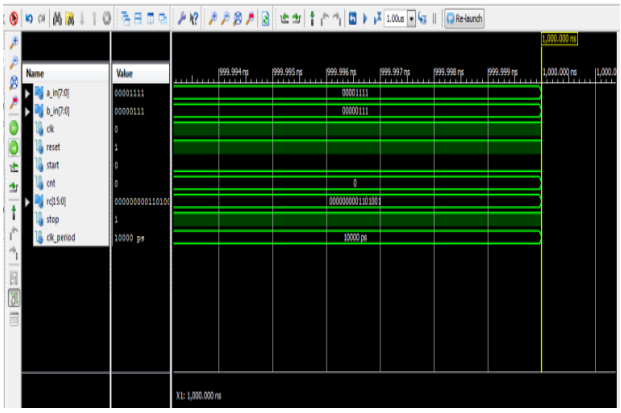
In fact, while the adder is busy with performing the addition, there is enough time for the ring counter and multiplexer M1 to deliver the value of the next hot bit. All delays in this path are shorter than the adder delay and, hence, do not increase the delay of BZ-FAD. The synthesis timing reports estimates the critical path delay for the BZ-FAD and the conventional multipliers to be 6.599 and 6.472 ns, respectively, which agrees with the above discussion. The slight difference between the reported delays originates from the fact that the input clock signal to the Feeder and Bypass registers pass through a NAND and a NOR gate in the BZ-FAD architecture.

V. Simulation results

Both the architectures , conventional shift and add multiplier and Bypass zero feed a direct multiplier are implemented in VHDL and are simulated using Xilinx in simulator and the results obtained are shown in following sections.

A. Conventional Multiplier output

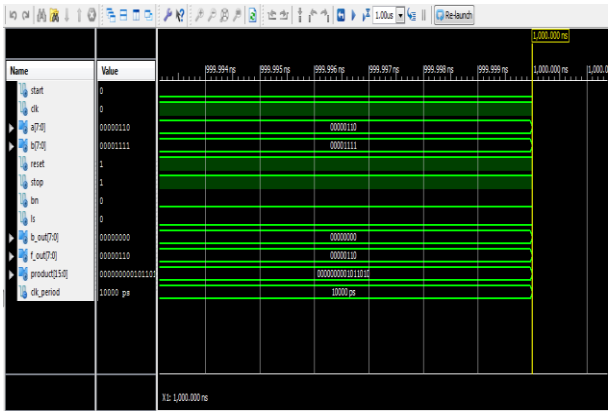
The simulation result for 8 bit multiplier using conventional architecture is highlighted in Figure 5.1. The inputs given are 00001111(15) and 000000111(7). Multiplier is shifted in each cycle and the corresponding partial products are formed. The conventional multiplier uses a binary counter. From the simulation results, it can be seen that the output of multiplier comes out to be 0000000001101001(105).



1. Simulation results of conventional multiplier

B. BZFAD multiplier output

The simulation result for 8 bit multiplier using low power architecture is shown in Figure 5.2. The inputs given are 00001111(15) and 00000110(6).The multiplication operation is performed using the BZFAD architecture. The low power multiplier uses a low power ring counter. From the simulation results, it can be seen that the output comes out to be 00000000011011010(90)



2. Simulation results of BZFAD multiplier

C. Device utilization summary

TABLE I DEVICE UTILIZATION SUMMARY OF CONVENTIONAL MULTIPLIERS

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	81	960	8%
Number of Slice Flip Flops	88	1920	4%
Number of 4 input LUTs	123	1920	6%
Number of bonded IOBs	54	83	65%
Number of GCLKs	1	24	4%

TABLE II Device utilization summary of BZFAD multiplier

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of Slice Flip Flops	55	1,920	2%
Number of 4 input LUTs	91	1,920	4%
Logic Distribution			
Number of occupied Slices	64	960	6%
Number of Slices containing only related logic	64	64	100%
Number of Slices containing unrelated logic	0	64	0%
Total Number of 4 input LUTs	123	1,920	6%
Number used as logic	91		
Number used as a route-thru	32		
Number of bonded IOBs	68	66	103%
IOB Flip Flops	8		
Number of BUFGMUXs	1	24	4%

D. Timing summary

1). Conventional multiplier minimum period= 6.599 ns
2). BZFAD multiplier minimum period= 6.472 ns.

TABLE III. Summary of Simulation Results

Multiplier type	Conventional	BZFAD
Vendor	Xilinx	Xilinx
Device and family	Spartan 3E	Spartan 3E
Power dissipation	34mW	27Mw

VII. Conclusion

In this project, sources of power dissipation in VLSI circuits are studied and methods to reduce power dissipation are explained. Switching activity is found to be active source of power dissipation in conventional shift and add architecture. A low power architecture known as Bypass Zero Feed A Direct (BZFAD) architecture is proposed. Both the architectures are implemented using VHDL in Xilinx and results thus obtained are analyzed. The proposed architecture reduced the power dissipated by 20% compared to conventional multiplier architecture.

VIII. Future Scope

The project can be implemented on CADENCE or SYNOPSIS tool to obtain more reliable power calculations. Multiplier architectures other than shift and add multiplier can be implemented, analyzed and compared with one another to select an appropriate architecture for our purpose. Apart from switching activity other sources of power dissipation can be considered and addressed.

References

1. Ercegovac M.D. and Huang Z. (March 2006) "High performance low power left to right array multiplier design" IEEE Trans. Comput., Vol-54, no-2, pp 272-283.
2. C. N.Marimuthu, Dr. P. Thangaraj, Aswathy Ramesan "Low power shift and add multiplier design" International Journal of Computer Science and Information Technology, Volume 2, No 3, June 2010.
3. Prof Prasann D.Kulkarni, Prof.S.P.Deshpande, Dr.G.R.Udupi" low power add and shift multiplier design Bzfad architecture" International Journal of Computer and Electronics Research [Vol 2, Issue 2, April 2013]
4. N.Y.Shen and O.T.C.Chen."Low power multipliers by minimizing switching activities of partial products" in Proc. IEEE Int.Symp.Circuits Syst., May 2002, Vol.4, pp 93-96.
5. Peiyi Zhao and Zhongfeng Wang,"Low Power design of VLSI circuits and systems" .ASIC,2009. ASICON ‟ 09. IEEE 8th International Conference.
6. A.Chandrakasan and R. Brodersen, "Low Power CMOS Digital Design", IEEE J. Solid State Circuits, Vol.27, no.4, pp 473-484, Apr 1992
7. M. Mottaghi-Dastjerdi, A. Afzali-Kusha, and M. Pedram,"BZ-FAD :A low power low area Multiplier based on Shift and Add architecture" IEEE transactions on very large scale integration(VLSI) systems, Vol.17, No.2, Feb 2009
8. O.T.Chen, S.Wang and Y, W.Wu "Minimization of switching activities of partial products for designing low power multipliers" IEEE Trans. Very Large Scale Integer (VLSI)Syst., Vol.11, No-3, pp418-433, June 2003
9. Zamin Ali Khana ,S. M. Aqil Burneyb, , Jawed Naseemc, Kashif Rizwand, "Optimization of Power Consumption in VLSI Circuit", IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 2, March 2011
10. K.H.Chen and Y.S.Chu , "A low power multiplier with spurious power suppression technique" ,IEEE Trans. Very Large Scale Integr .(VLSI)Syst. , Vol.15 , no-7,pp846-850, July 2007.
11. V. P. Nelson, H. T. Nagle, B. D. Carroll, and J. I. David,"Digital Logic Circuit Analysis & Design." Englewood Cliffs, NJ: Prentice-Hall, 1996.